Smart Round Robin CPU Scheduling Algorithm For Operating Systems

Qaleel Sha Backer
Department of Computer Science,
Lovely Professional University.
Jalandhar, Punjab, India

Amal C
Department of Computer Science,
Lovely Professional University
Jalandhar, Punjab, India

Vivek Kumar
Department of Computer Science,
Lovely Professional University
Jalandhar, Punjab, India

Ravindra Singh Yadav
Supervisor
Department of Computer Science,
Lovely Professional University
Jalandhar, Punjab, India

Abstract— **Multitasking is one of the key aspects in operating systems. CPU scheduling helps in improving the multitasking efficiency of an operating system. An operating system has to manage many system resources. CPU time is the most important System Resource. A multitasking operating system has many processes each requiring CPU time. Operating system needs to manage the CPU time. It is responsible for assigning CPU time to each process. A common method for CPU scheduling is a Round Robin Scheduling method. It is a preemptive CPU scheduling method which uses a predefined time quantum to execute processes in the ready queue of the operating system**

This paper proposes a new method where the time quantum changes every cycle depending on the processes remaining time. A variable time quantum gives processes an option to take additional quantum time. The collaboration of the two, forms the Smart Round Robin. It produces less Average Waiting Time (AWT) and less average Turnaround Time (TAT) in comparison to the Traditional Round Robin Scheduling method.

Keywords—CPU scheduling, Round Robin CPU scheduling algorithm, Turnaround time, Waiting time, Response time, Time quantum, Gantt chart, Operating System.

I. INTRODUCTION

The processes are allocated system resources through operating system. The most important resource is CPU time among other resources. Allocating CPU time to respective processes is called CPU scheduling. Scheduling is a fundamental function of operating system that identifies the priority of the process, when there are multiple executable processes waiting to execute and allocates CPU time to them. CPU scheduling is an important aspect of the operating system due to variety of processes that require CPU time. It affects the system performance. There exists number of CPU scheduling algorithms like the First Come First Serve [FCFS], Shortest Job First [SJF], Round Robin [1] etc.

CPU scheduling means the algorithm to select the next process for execution. And during CPU scheduling context switches occur. Context switches means that the process in execution is stopped and another process is provided with CPU time. This reduces the performance of the CPU. To optimize the CPU usage, an efficient scheduling algorithm is a must. Different CPU scheduling algorithms have different properties and may favor one system metric over another. While selecting the better algorithm in a particular situation, we must consider the different properties and performance attributes of that algorithm. CPU efficiency is an important criterion while selecting a process for execution. The CPU should never be left idle. So, it is necessary that CPU scheduling is done in a way that the system is used most effectively and systematically [2]. There are three types of schedulers, a long-term scheduler, a mid-term scheduler and a short-term scheduler. The longterm scheduler is responsible for selecting a process from the queue and loading it for execution. The short -term scheduler selects one of the tasks that is ready to execute and allocates them CPU time. The mid-term scheduler removes the less used or dormant processes from the memory to reduce the degree of multiprogramming [11]. The throughput is an important measure for CPU scheduling. It is the number of processes that are completed per unit time.

The time interval from the instant a process has been submitted till its competition is the turnaround time. The CPU scheduling algorithm does not affect the amount of time during which a procedure is being executed or does I/O execution. It affects only the amount of time that a process has to wait in the ready queue. Waiting time is the sum of such periods spent waiting in the ready queue. Another measure is the time from the submission of a request until the first response is produced. This measure, known as response time, is the time that it takes to start responding [3].

A context switch is the action of storing state of the executing process and restoring state of a preempted process, so that execution can be resumed from the same point at a later time. Context switching is usually computationally intensive, leading to wastage of time and memory. This time is usually an overhead to the scheduler and the operating system [4].

The algorithms proposed by M Ramakrishna and G. Pattabhi Rama Rao in [5] and Ankush Joshi and Shubhashish Goswami in [8], has a fixed time quantum for all cycles. In Optimized Round Robin Scheduling algorithm proposed by Ajit Singh, Priyanka Goyal and Sahil Batra [4], the time quantum is dynamic but is not computed based on the burst times of the processes. A process having very small remaining time (say 1 ms) will have to wait for an entire cycle

to get the CPU time and complete execution. The need for a better scheduling algorithm arises to resolve these problems. The Smart Round Robin aims at clearing these issues and reducing the average waiting time (AWT) and average turnaround time (ATAT) of the processes.

Round robin scheduling algorithm in real world applications, finds its use mainly in process scheduling and network schedulers for packet scheduling. In network schedulers, the packets are grouped and sent over a network that employs round robin scheduling algorithm. A multiplexer has unique queues for each data flow. The round robin algorithm allows every active queue to take turns in transferring packets on a multiplexed channel. It provides fairness to all queues as the queue that has waited for the longest time gets the highest priority. But in case of unequal size of data packets, it may fail to provide fairness to all the queues. Queues with larger data packets would be favored over others [9]. A need for a smarter round robin arises to provide dynamic time quantum based on the size of the data packets in each queue. Smart Round Robin algorithm does exactly that by computing a new time quantum every cycle based on the remaining burst time of the tasks.

II. Related Work

In recent times different approaches have been used to increase the performance of CPU scheduling algorithms. Neha Mittal, Khushbu Garg and Ashish Ameria proposed a new algorithm, the Modified Round Robin algorithm [3]. In the given algorithm, the processes are arranged in increasing order of their burst times. The modified time quantum is calculated based on the number of processes. The time quantum is applied to all processes. Figure 1. represents the Modified Round Robin algorithm steps. The advantage of this method is that the time quantum more efficiently rolls around the processes reducing its waiting time. The drawback is that for every cycle the time quantum is static so it does not provide dynamicity for each cycle.

M Ramakrishna and G. Pattabhi Rama Rao worked on an efficient Round Robin [5]. It integrates Priority scheduling and traditional Round Robin scheduling algorithms. Each process gets a fixed priority, so the processes are executed methodically but the drawback is that the time quantum is pre-defined and fixed so it eliminates the dynamic nature of the algorithm.

Sanjaya Kumar Panda and Sourav Kumar Bhoi, proposed a Min-Max Round robin (MMRR) algorithm [12]. The time quantum is dynamically calculated for each cycle as the difference of maximum remaining time and minimum remaining time. The algorithm does not allow completion of such processes which need very short time within the same cycle.

Lipika Datta made an efficient Round Robin [7]. It is based on dynamic time slice. It has the dynamicity of time quantum but it is complex to perform.

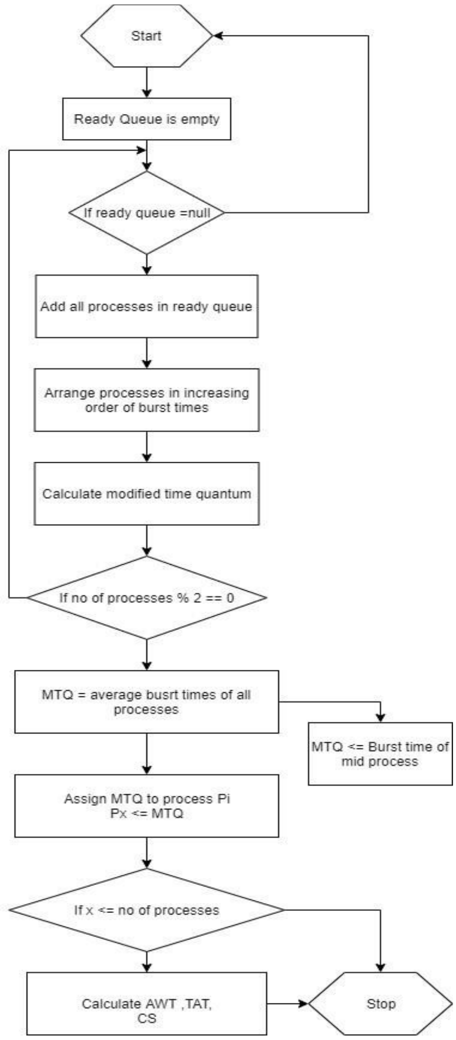

Fig 1. Modified Round Robin Algorithm [3]

Amar Ranjan Dash, Sandipta kumar Sahu and Sanjay Kumar Samantra proposed an Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum [6]. Their proposed algorithm is Dynamic Average Burst Round Robin (DABRR) finds a new time quantum for each cycle providing dynamicity. The time quantum for a cycle is fixed and cannot be changed even if the task requires very less CPU time for completion. Such tasks will have to wait for another cycle before it gets the CPU time to complete execution. Figure 2. Represents the DABRR Algorithm.

TQ: Time Quantum RQ: Ready Queue n: number of processes TBT: Total Burst Time

Pi: Process at ith index i, j: indexes for the ready queue

1. The processes are arranged in ascending order of their burst times.
2. n= number of processes in RQ.
3. Initialize i=0; TBT=0.

4. Repeat step 5 and 6 till i < n
5. TBT += burst time of process Pi
6. i++
7. TQ = TBT/n
8 j = 0
9. Repeat from step 12 to 19 till j<n
10. if (burst time of Pi) <= TQ
11. Execute the process
12. Take the process out of RQ
13. n-- 14. Else
15. Execute the process for a time interval up to 1 TQ
16. Burst time of Pi = Burst time of Pi – TQ 17. Add the process to ready queue for next round of execution
18. j++
19. If new process arrives
20. goto step 1
21. If RQ is not empty
22. goto step 2

Fig 2. DABRR algorithm [6]

Ankush Joshi and Shubhashish Goswami proposed a Modified Round Robin algorithm using Priority Scheduling method [8]. They called it the Modulo Based Round Robin Algorithm. The algorithm lacks the dynamicity of the time quantum. Figure. 3 represents the flowchart of the proposed algorithm. The algorithm computes two values:

a) P = average CPU burst of all processes.
b) Mi = Burst for that process % P

The processes are then arranged in ascending order of M and time quantum P is applied to each process.

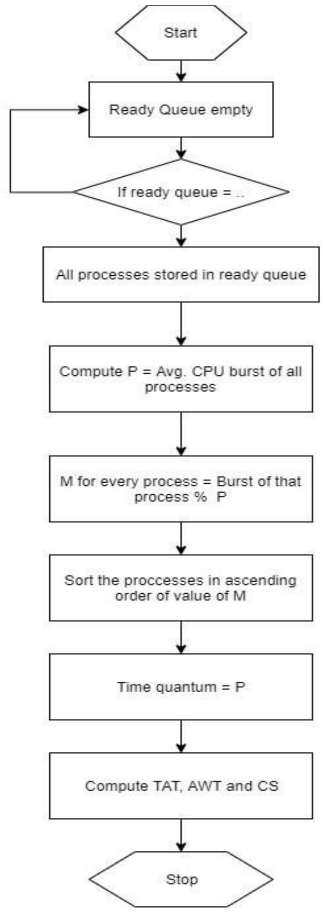

Fig 3. Flowchart for Modulo Based Round Robin Algorithm [8]

III. THE PROPOSED ALGORITHMM

The Smart Round Robin CPU Scheduling algorithm focuses on the elementary criteria of CPU Scheduling. We assume that all the processes have a predefined burst time. The processes are first put in the ready queue, waiting to be executed. Every cycle the processes are arranged in increasing order of their remaining time. For the first pass the remaining time is equal to the burst time.

The time quantum is dynamic throughout the cycles. A new Smart Time Quantum (STQ) and Delta for each cycle is computed. If a process can complete its execution within (STQ + Delta) time, it is allowed to finish its execution. This lets tasks with very less remaining time to finish its execution rather than wait for an entire cycle to acquire very short CPU time. The STQ and Delta together provide flexibility to the processes to execute dynamically and take extra CPU time to reduce the waiting time. This makes the algorithm very efficient and increases CPU output.

3.1 Steps for Smart Round Robin

1) Start.
2) Empty The ready Queue.
3) Change the status of all the processes to ready state.
4) Arrange the processes in ready queue in increasing order of their remaining burst time (RBT).

For the first cycle RBT = BT

5) Calculate the difference between adjacent RBT. Smart Time Quantum (STQ) is the average of those differences. Round off the value to the nearest integer.

STQ = Avg. of the differences of adjacent RBT

TABLE I. An example to calculate STQ.

Process ID	Remaining burst time (RBT)
P0	6
P1	9
P2	16

From the example in table I. the difference in adjacent RBT is 3 and 7 respectively. STQ = avg (3,7) = 5 ms.

6) Delta value is computed to be half of the Smart Time Quantum (STQ) rounded off to lower integer value (Floor Rounding).

Delta = STQ/2

For the example in table I. Delta = floor (5/2) = 2 ms. *7)* For each process in the cycle: IF RBT <= (STQ + Delta)) then

Assign RBT ms on the CPU

Else

Assign STQ ms on CPU

8) Repeat steps 4) to 7) until all processes finish execution.

9) Calculate the Turnaround Time (TAT), Waiting Time (WT), Average Turnaround Time (ATAT) and Average Waiting Time (AWT). *10)* Stop.

As seen in Fig. 4, processes are sorted in the increasing order of RBT, this improves the efficiency of the CPU as tasks with shorter burst times do not have to wait for a long period of time.

In case of batch of tasks with different arrival times, the processes already in the ready queue gets CPU time according to the Smart Round Robin algorithm. If a process arrives later i.e. while a cycle is in execution it has to wait till the end of that cycle. After the end of that cycle the process is put into the ready queue available for selection.

3.2 Assumptions

1. Each process is CPU bound.
2. Priority of processes is not considered.
3. Processes do not depend on each other and do not have a deadline.
4. Burst times, arrival times and number of processes are known beforehand.
5. Context switching overhead is ignored when calculating the TAT, WT, ATAT, and AWT.

3.3 Flowchart

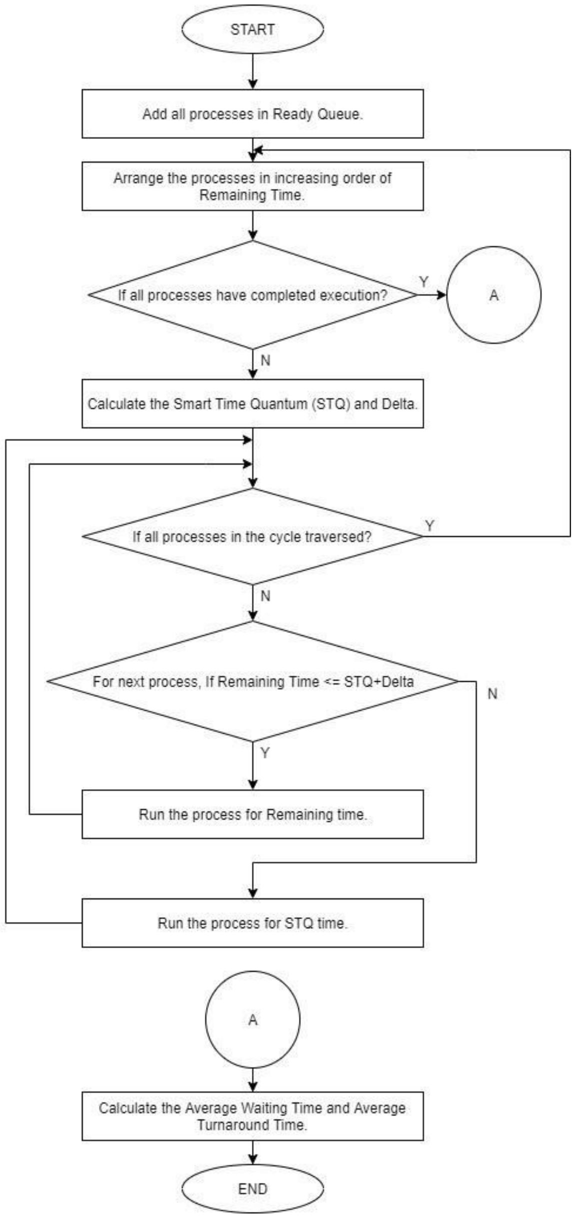

Fig 4. Flowchart for Smart Round Robin CPU Scheduling Algorithm.

3.4 Test Cases

3.4.1 Case I (General Case)

In case 1, we assume four processes, all arriving at time 0 ms. A random burst time is assigned to each process. For the traditional Round Robin, time quantum of 6 ms is assumed and the results are calculated and compared. All units are considered in milliseconds.

TABLE II. Case I (General Case)

Case 1		
Process	Arrival Time	Burst time
P0	0	12
P1	0	34
P2	0	8

P3	0	19

3.4.1.1 Analysis

With reference to Table II

P0	P1	P2	P3	P0	P1	P2	P3	P1	P3	P1	P3	P1

0 6 12 18 24 30 36 38 44 50 56 62 63 73

Fig 5. Gantt Chart for Traditional Round Robin Algorithm.

With reference to Algorithm in 3.1 and Table II

P2	P0	P3	P1	P3	P1

0 8 20 29 38 48 73

Fig 6. Gantt Chart for Smart Round Robin CPU Scheduling Algorithm.

Time Quantums for Smart Round Robin are calculated to be 9 ms and 15 ms, while the Delta values for those cycles are computed to be 4 ms and 7 ms respectively.

Method	Average Turnaround time	Average Waiting Time
Traditional Round Robin (Q=6)	51	32.75
Smart Round Robin	37.25	19
** All time units in ms		
Graphical Comparision of the data:		

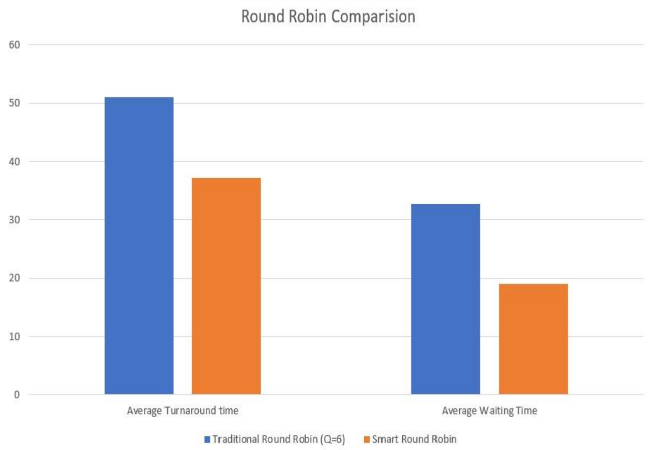

Fig 7. Comparison of Average Turnaround Time and Average Waiting Time for Table II

3.4.2 Case II (Small Burst Times)

In the following case, we have considered five processes arriving at time 0 ms, and having very short burst times. For the regular Round Robin, Time Quantum of 4 ms is assumed and the results are calculated. All units are considered in milliseconds.

TABLE III. Case II (Small Burst Times)

Case 2		
Process	Arrival Time	Burst time
P0	0	2
P1	0	5
P2	0	6
P3	0	3
P4	0	9

3.4.2.1 Analysis

With reference to Table III

P0	P1	P2	P3	P4	P1	P2	P4

0 2 6 1O 13 17 18 20
25

Fig 8. Gantt Chart for Traditional Round Robin Algorithm.

With reference to Algorithm in 3.1 and Table III

P0	P3	P1	P2	P4	P1	P2	P4	P2	P4

0 2 5 7 9 11 14 16 18 20 25

Fig 9. Gantt Chart for Smart Round Robin CPU Scheduling Algorithm.

Time Quantums for Smart Round Robin are calculated to be 2 ms, 2 ms and 3 ms, while the Delta values for those cycles are computed to be 1 ms for all three cycles.

Method	Average Turnaround time	Average Waiting Time
Traditional Round Robin (Q=4)	15.6	10.6
Smart Round Robin	13.2	8.6
** All time units in ms		
Graphical Comparision of the data:		

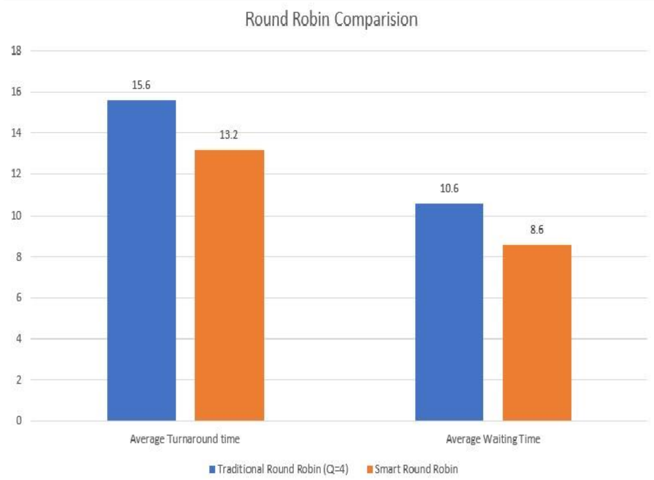

Fig 10. Comparison of Average Turnaround Time and Average Waiting Time for Table III

3.4.3 Case III (Small and Large Burst Times) In the following case, we have considered four processes arriving at time 0 ms, and having huge difference in burst times

signifying a batch of short and large processes. For the regular Round Robin, Time Quantum of 20 ms is assumed and the results are calculated. All units are considered in milliseconds.

TABLE IV. Case III (Small and Large Burst Times)

Case 3		
Process	Arrival Time	Burst time
P0	0	26
P1	0	67
P2	0	82
P3	0	11

3.4.3.1 Analysis

With reference to Table IV

PO	P1	P2	P3	P0	P1	P2	P1	P2	P1	P2

0 20 40 60 71 77 97 117 137 157 164 186

Fig 11. Gantt Chart for Traditional Round Robin Algorithm.

With reference to Algorithm in 3.1 and Table IV

P3	P0	P1	P2	P1	P2	P1	P2	P1	P2

0 11 37 61 85 100 115 130 145 158 186

Fig 12. Gannt Chart for Smart Round Robin CPU Scheduling Algorithm.

Time Quantums for Smart Round Robin are calculated to be 24 ms ,15 ms, 15 ms, and 15 ms for each cycle respectively, while the Delta values for those cycles are computed to be 12 ms, 7 ms, 7 ms, and 7 ms respectively.

Method	Average Turnaround time	Average Waiting Time
Traditional Round Robin (Q=20)	124.5	78
Smart Round Robin	98	51.5

** All time units in ms

Graphical Comparision of the data:

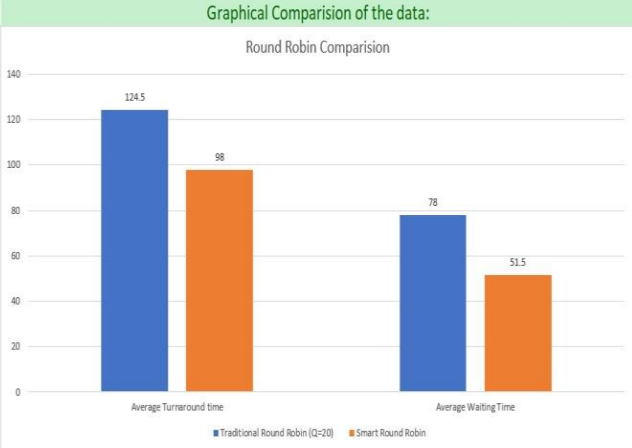

Fig 13. Comparison of Average Turnaround Time and Average Waiting Time for Table IV

3.4.4 Case IV (Different Arrival Time)

Here, we have assumed four processes, all arriving at different times. A random burst time is assigned to each process. For the regular Round Robin, Time Quantum of 2 ms is assumed and the results are calculated. All units are considered in milliseconds.

TABLE V. Case IV (Different Arrival Time)

Case 4		
Process	Arrival Time	Burst time
P0	0	8
P1	2	6
P2	7	11
P3	0	5

3.4.4.1 Analysis

With reference to Table V

P0	P3	P0	P1	P3	P0	P1	P2	P3	P0	P1	P2

0 2 4 6 8 10 12 14 16 17 19 21 30

Fig 14. Gantt Chart for Traditional Round Robin Algorithm.

With reference to Algorithm in 3.1 and Table V

P3	P0	P3	P0	P1	P0	P1	P2

0 3 6 8 10 12 15 19 30

Fig 15. Gannt Chart for Smart Round Robin CPU Scheduling Algorithm.

Time Quantums for Smart Round Robin are calculated to be 3 ms, 2 ms, and 4 ms for each cycle respectively, while the Delta values for those cycles are computed to be 2 ms,1 ms, and 2 ms respectively.

Method	Average Turnaround time	Average Waiting Time
Traditional Round Robin (Q=2)	19.5	12.25
Smart Round Robin	12.75	8.25

** All time units in ms

Graphical Comparision of the data:

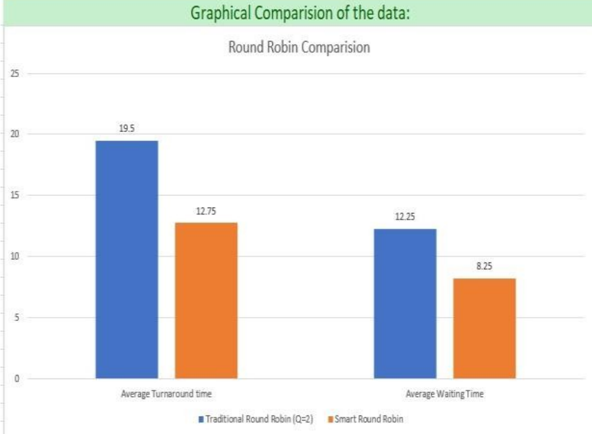

Fig 16. Comparison of Average Turnaround Time and Average Waiting

Time for Table V

IV. Analysis And Result

The Smart Round Robin performs better than the Traditional Round Robin CPU scheduling algorithm in terms of ATAT and AWT. Table VI. shows the % reduction in the parameters when Smart Round Robin is used.

TABLE VI. Reduction in ATAT and AWT due to Smart Round Robin

Parameters	% Reduction in ATAT	% Reduction in AWT
Case I (general case)	26.96	41.98
Case II (small burst Times)	15.38	18.86
Case III (small and large burst times)	21.28	33.97
Case IV (Different arrival time)	34.615	32.65

From Table VI. it can be calculated that the algorithm reduces the ATAT by 24.559% on average and AWT by 31.865 % on average compared to the Traditional Round Robin.

Figure 17. and figure 18. Graphically represents the % reduction in ATAT and AWT respectively.

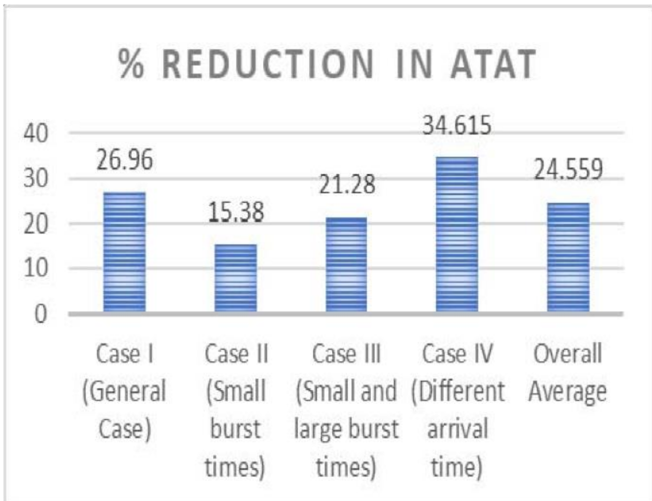

Fig 17. Percentage reduction in ATAT from table VI.

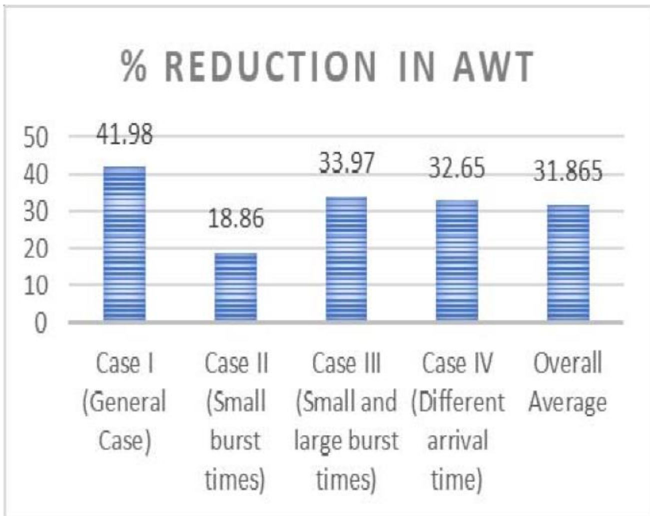

Fig 18. 1Percentage reduction in AWT from table VI.

The Smart Round algorithm is further compared along side the Optimized Round Robin algorithm [4], Efficient Round Robin algorithm [5] and Modulo based Round Robin algorithm [8]. The results for each algorithm for the test cases I, II and III from tables II, III and IV are computed. For the Modulo based Round Robin algorithm [8] the task with lowest burst time is considered to be of the highest priority. Table VII. compares the algorithm's ATAT and table VIII. compares the algorithm's AWT. All units are considered in milliseconds.

TABLE VII. Comparison of algorithms in terms of ATAT.

Parameters	Case I (General case)	Case II (Small burst times)	Case III (Small and large burst times)
Optimized Round Robin [4]	47	15.6	124.5
Efficient Round Robin [5]	44	13.6	99.5
Modulo based Round Robin [8]	53	15.6	96.25
Smart Round Robin	37.25	13.2	98

TABLE VIII. Table VIII. Comparison of algorithms in terms of AWT.

Parameters	Case I (General case)	Case II (Small burst times)	Case III (Small and large burst times)
Optimized Round Robin [4]	28.75	10.6	78
Efficient Round Robin [5]	25.75	9	53
Modulo based Round Robin [8]	34.75	10.6	49.75
Smart Round Robin	19	8.6	51.5

The results in table VII. and VIII. have been represented graphically in figure 19. and figure 20.

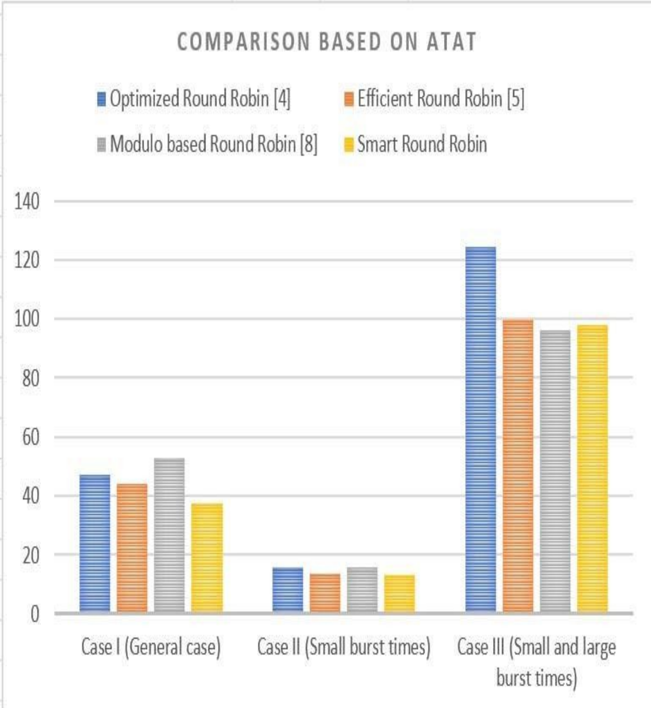

Fig 19. Comparison chart based on ATAT.

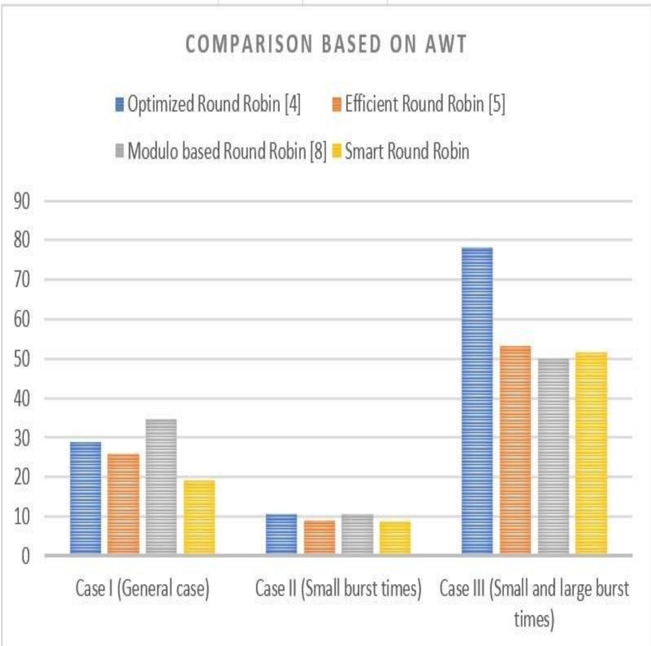

Fig 20. Comparison chart based on AWT.

From figure 17. and figure 18. it is clear that the Smart Round CPU Scheduling algorithm outperforms other algorithms by reducing the ATAT and AWT in almost every case.

V. Conclusion and Future Work

The main objective of CPU Scheduling is to improve the system efficiency. From section IV and V, it is concluded that the Smart Round Robin proposed in this paper outperforms Traditional Round Robin and other algorithms in terms of AWT and ATAT. The algorithm provides dynamic time quantum for each cycle and specifically helps tasks which have short remaining burst times.

Future work can be based on modifying and implementing the algorithm for multiprocessor systems and real time systems where processes might have deadlines and must be completed within that time constraint. In future, priority of each task can be considered as an additional information along with process ID, burst time and arrival time. Based on the priority, lower priority processes will be preempted if a high priority task requires CPU time.

References

[1] Silberschatz, A., Peterson, J. L., and Galvin, B.,Operating System Concepts, Addison Wesley, 7th Edition, 2006.

[2] E.O. Oyetunji, A. E. Oluleye," Performance Assessment of Some CPU Scheduling Algorithms", *Research Journal of Information Technology*, 1(1): pp 22-26, 2009.

[3] N. Mittal, K. Garg, A. Ameria , (2015) "A Paper on Modified Round Robin Algorithm", *IJLTEMAS*, Volume 4, Issue 11, ISSN 2278 – 2540.

[4] A. Singh, P. Goyal, S. Batra, (2010) " An Optimized Round Robin Scheduling Algorithm for CPU Scheduling", *(IJCSE) International Journal on Computer Science and Engineering,* Vol. 02, Issue 7, 2383-2385.

[5] M. Ramakrishna , G. Pattabhi Rama Rao, (2013) "Efficient Round Robin CPU Scheduling Algorithm For Operating System " *IJITR*, Vol. 1 , Issue 1, 103-109.

[6] A. R. Dash, S. K. Sahu, S. K. Samantra, (2015) "An Optimised Round Robin CPU Scheduling Algoith with Dynamic Quantum Time ", *IJSCEIT* , Vol. 5, Issue 1.

[7] L. Datta, (2015) " Efficient Round Robin Scheduling Algorithm with Dynamic Time Slice " , *IJEME*, Vol 5, Issue 2.

[8] A. Joshi, S. Gosswami, (2017) "Modified Round Robin algorithm by using Priority Scheduling ", *Advances in Computational Sciences and Technology*, ISSN 0973-6107 Volume 10, pp. 1543-15.

[9] Round Robin Scheduling Algorithm wikipedia: https://en.wikipedia.org/wiki/Round-robin_scheduling

[10] Operating Systems by Sibsankar Haldar 2009, Pearson Education India.

[11] A. Noon, A. Kalakech, S. Kadry, "A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average", *IJCSI International Journal of Computer Science Issues*, Vol. 8, Issue 3, No. 1, May 2011.

[12] S. K. Panda, & S. Bhoi. (2014). An Effective Round Robin Algorithm using Min-Max Dispersion Measure. *International Journal*. 4.

[13] Philipe A. Jansen, "Operating Systems / Structures and Mechanisms" pages 77-80, Academic Press, Inc., 198.

[14] William Stallings, Ph.D., "Operating Systems / Internals and Design Principles", pages 75-76, 406-408, Prentice Hall, 2001.

[15] Englander, I., 2003. The Architecture of Computer Hardware and Systems Software; An Information Technology Approach, 3rd Edition, John Wiley & Sons, Inc.

[16] I. E. W. Giering and T. P. Baker, "A tool for the deterministic scheduling of real-time programs implemented as periodic Ada tasks," Ada Lett., vol. XIV, pp. 54-73, (1994).